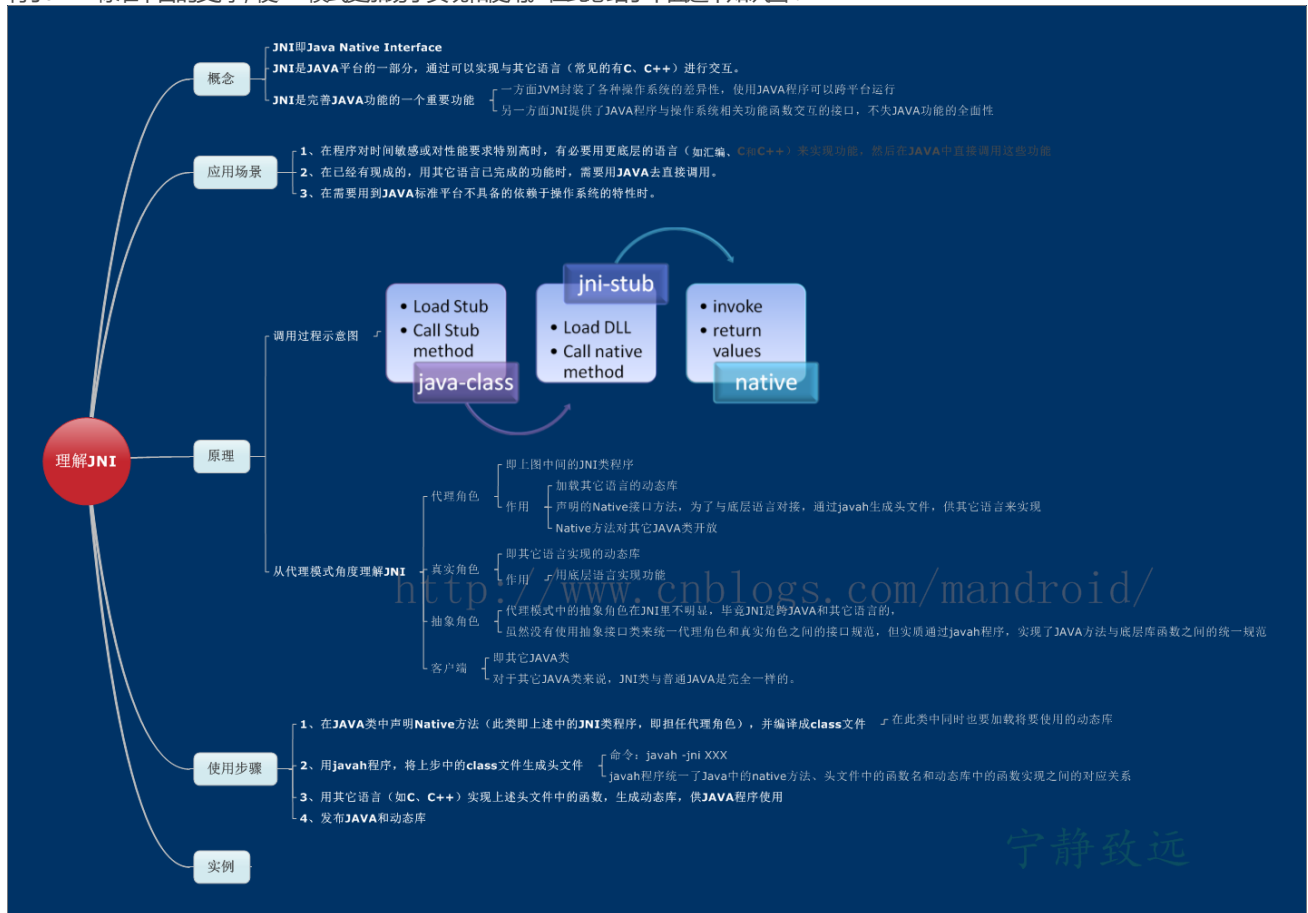


JAVA基础之理解JNI原理 - 宁静致远 - 博客园

JNI是JAVA标准平台中的一个重要功能，它弥补了JAVA的与平台无关这一重大优点的不足，在JAVA实现跨平台的同时，也能与其它语言（如C、C++）的动态库进行交互，给其它语言发挥优势的机会。
有了JAVA标准平台的支持，使JNI模式更加易于实现和使用。在此总结了下面这个知识图：



实例：

环境说明：ubuntu 10.4.2 LTS系统

程序清单1：src/com/magc/jni/HelloWorld.java



```
1/**
2 *
3 */
4package com.magc.jni;
5
6/**
7 * @author magc
8 *
9 */
10publicclass HelloWorld {
11
12static {
13
14 System.loadLibrary("Hello");
15
16 }
17
18publicnativevoid DisplayHello();
19/**
20 * @param args
21 */
22publicstaticvoid main(String[] args) {
23
24new HelloWorld().DisplayHello();
25 }
26
27}
```



进入src目录下，编译刻JAVA类，

命令：javac ./com/magc/jni/HelloWorld.java

在刻HelloWorld.java所在目录下生成HelloWorld.class

然后使用javah生成头文件，

命令：javah -jni com.magc.jni.HelloWorld

在当前目录下生成com_magc_jni_HelloWorld.h头文件，此文件供C、C++程序来引用并实现其中的函数
程序清单2：com_magc_jni_HelloWorld.h



```
1/* DO NOT EDIT THIS FILE - it is machine generated */
2#include <jni.h>
3/* Header for class com_magc_jni_HelloWorld */
4
5#ifndef _Included_com_magc_jni_HelloWorld
6#define _Included_com_magc_jni_HelloWorld
7#ifdef __cplusplus
8extern"C" {
9#endif
10/*
11 * Class: com_magc_jni_HelloWorld
12 * Method: DisplayHello
13 * Signature: ()V
14*/
15JNIEXPORT void JNICALL Java_com_magc_jni_HelloWorld_DisplayHello
16 (JNIEnv *, jobject);
17
18#ifdef __cplusplus
19}
20#endif
21#endif
```



注：1)、此头文件是不需要用户编译的，直接供其它C、C++程序引用。

2)、此头文件中的Java_com_magc_jni_HelloWorld_DisplayHello (JNIEnv *, jobject)方法，是将来与动态链接库交互的接口，并需要名字保持一致。

程序清单3：src/jni_helloworldImpl.cpp



```
#include <jni.h>
#include "com_magc_jni_HelloWorld.h"
#include <stdio.h>
JNIEXPORT void JNICALL Java_com_magc_jni_HelloWorld_DisplayHello
(JNIEnv *env, jobject obj)
{
    printf("From jni_helloworldImpl.cpp :");
    printf("Hello world ! \n");
    return;
}
```



此C++文件实现了上述头文件中的函数，注意方法函数名要保持一致。

编译生成动态库libHello.so

命令：g++ -shared -fPIC -I /usr/lib/jvm/java/include -I /usr/lib/jvm/java/include/linux jni_helloworldImpl.cpp -o libHello.so

成功后，便会在当前目录下生成动态链接库libHello.so文件。

有了具体实现的动态库后，就可以运行JAVA调用JNI程序类的native方法了，

命令：java -Djava.library.path=. com.magc.jni.HelloWorld

输入结果即为：From jni_helloworldImpl.cpp :Hello world !

总结：

Linux下

1. 创建文件 src/com/magc/jni/HelloWorld.java
2. 编译类 javac ./com/magc/jni/HelloWorld.java
3. javah生成头文件 javah -jni com.magc.jni.HelloWorld
4. 创建jni_helloworldImpl.cpp 实现com_magc_jni_HelloWorld.h头文件中的方法
5. 生成动态库 g++ -shared -fPIC -I /usr/lib/jvm/java/include -I /usr/lib/jvm/java/include /linux jni_helloworldImpl.cpp -o libHello.so
6. 使用Java调用JNI程序类的native方法
java -Djava.library.path=. com.magc.jni.HelloWorld

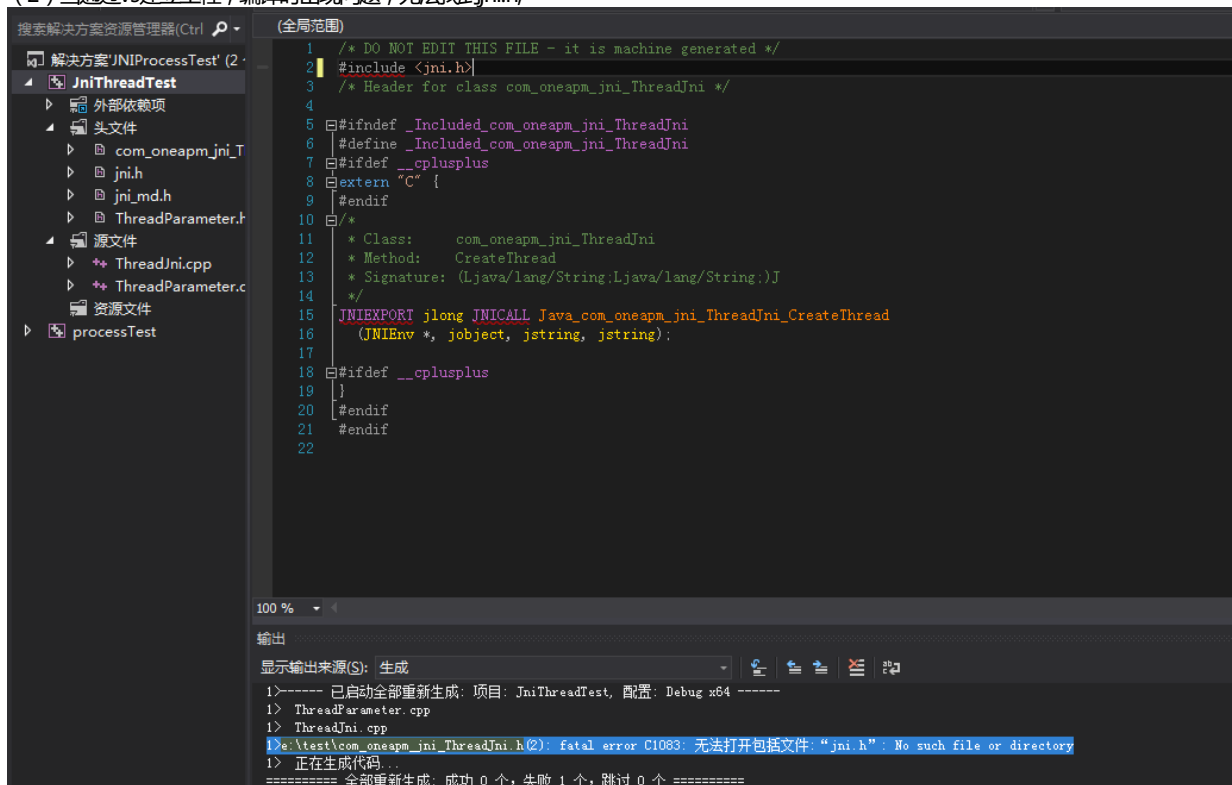
Windows下：

1. 创建文件 src/com/magc/jni/HelloWorld.java
2. 编译类 javac ./com/magc/jni/HelloWorld.java
3. javah生成头文件 javah com.magc.jni.HelloWorld
4. 创建jni_helloworld.cpp 实现com_magc_jni_HelloWorld.h头文件中的方法
5. 生成动态库 :有两种方式生成dll
 - (1) 通过cl命令, vs开发人员命令提示工具打开cmd
cl -I %JAVA_HOME%\include -I %JAVA_HOME%\include\win32 -LD helloworld.cpp -Fehello.dll
 - (2) 通过vs建立工程, 生成dll
6. 使用Java调用JNI程序类的native方法
在系统变量path中添加hello.dll所在为路径
java com.magc.jni.HelloWorld

问题:

windows下生成dll有些问题

- (1) 当通过cl命令生成时, 我的机器生成的是32bit dll, 而我的机器是64位的, 暂时还未解决!!
- (2) 当通过vs建立工程, 编译时出现问题, 无法找到jni.h,



The screenshot shows the Visual Studio IDE. On the left, the Solution Explorer displays a project named 'JniThreadTest' with files like 'jni.h', 'jni_md.h', and 'ThreadParameter.h'. The main editor window shows the content of 'jni.h', which includes standard JNI headers and a function declaration for 'Java_com_oneapm_jni_ThreadJni_CreateThread'. The bottom output window shows the build process, with a fatal error C1083 reported: '无法打开包括文件: "jni.h": No such file or directory'.

```
1  /* DO NOT EDIT THIS FILE - it is machine generated */
2  #include <jni.h>
3  /* Header for class com_oneapm_jni_ThreadJni */
4
5  #ifndef _Included_com_oneapm_jni_ThreadJni
6  #define _Included_com_oneapm_jni_ThreadJni
7  #ifdef __cplusplus
8  extern "C" {
9  #endif
10 /*
11  * Class:      com_oneapm_jni_ThreadJni
12  * Method:     CreateThread
13  * Signature:  (Ljava/lang/String;Ljava/lang/String;)J
14  */
15  JNIEXPORT jlong JNICALL Java_com_oneapm_jni_ThreadJni_CreateThread
16      (JNIEnv *, jobject, jstring, jstring);
17
18 #ifdef __cplusplus
19 }
20 #endif
21 #endif
22
```

输出

显示输出来源(S): 生成

```
1>----- 已启动全部重新生成: 项目: JniThreadTest, 配置: Debug x84 -----
1> ThreadParameter.cpp
1> ThreadJni.cpp
D:\test\com_oneapm_jni_ThreadJni.h(2): fatal error C1083: 无法打开包括文件: "jni.h": No such file or directory
1> 正在生成代码...
===== 全部重新生成: 成功 0 个, 失败 1 个, 跳过 0 个 =====
```

